Sisalema Yostin Ariel

Roy Playment System

METHODS:

1.MainWindow

1.1 void Main(String[] args)

2.UserInterface

2.1. void showMainMenu()

2.2. void viewEmployee()

2.3. void addEmployee()

2.4. void editEmployee()

2.5. void deleteEmployee()

2.6. String getUserInput(String errorMessage)

2.7. boolean getBooleanInput(String errorMessage)

2.8. int getIntInput(String errorMessage)

3.EmployeeManager

3.1. void EmployeeManager()

3.2. void addEmployee(Employee employee)

3.3. void updateEmployee(int index,Employee employee)

3.4. void removeEmployee(int index)

3.5. List<Employee> getEmployees()

3.6. List<Employee> loadEmployees()

3.7. void saveEmployees()

4.LoginScreen

4.1. void registerUser

4.2. boolean checkPassword()

5.PayrollGenerator

5.1. void generatePayrolls(List<Employee> employees)

5.2. IndividualPayroll generateIndividualPayroll(Employee employee,Date startMonthDate,Date endMonthDate)

5.3. List<GeneralPayroll> generateGeneralPayroll(List<Employee> employees)

5.4. JsonObject savePayrollsToFile(Json payrolls)

6.IndividualPayroll

6.1. double calculateTotalIncome()

7.IndividualPayroll

7.1. Date getStartMonthDate()

7.2. Date getEndMonthDate()

8.Income

8.1. double calculateTotalExpenses()

9.Calculator

9.1. double calculateBasicSalary(double annualSalary)

9.2. double calculateOvertimeHours(double hoursWorked, double regularHours, double hourlyRate)

9.3. double calculateReserveFunds(double basicSalary)

9.4. double calculateLessContribution(double totalIncome, double reserveFunds)

9.5. double calculateBiweeklyAdvance(double basicSalary)

9.6. double calculateFoodDeduction(boolean bringsOwnFood, double foodDeductionAmount)

9.7. double calculateTotalIncome(double basicSalary, double overtimePayment, double bonuses)

9.8. double calculateTotalExpenses(double iessContribution, double biweeklyAdvance, double lessLoans, double companyLoans, double fines, double foodDeduction)

9.9. double calculateNetPayment(double totalIncome, double totalExpenses)

9.10. double calculateEmployerContribution(double totalIncome, double reserveFunds)

9.11. double calculateTotalEmployeeCost(double netPayment, double employerContribution)

TEST CASES:

Calculator

9.1. calculateBasicSalary

Input: annualSalary

Output: double

annualSalary expectedOutput

9.1.1 12000 1000.00

9.1.2 24000 2000.00

9.1.3 0 0.00

9.2. calculateOvertimeHours

Input: hoursWorked,regularHours,hourlyRate

Output: double

hoursWorked regularHours hourlyRate expectedOutput

9.2.1 50 40 10 150.00

9.2.2 45 40 15 112.50

9.2.3 40 40 20 0.00

9.3. calculateReserveFunds

Input: basicSalary

Output: double

basicSalary expectedOutput

9.3.1 1000 83.30

9.3.2 2000 166.60

9.3.3 0 0.00

9.4. calculateLessContribution

Input: totalIncome,reserveFunds

Output: double

totalIncome reserveFunds expectedResult

9.4.1 2500.0 200.0 217.35

9.4.2 5000.0 400.0 434.70

9.4.3 0.0 0.0 0.0

9.5. calculateBiweeklyAdvance

Input: basicSalary

Output: double

basicSalary expectedResult

9.5.1 2000.0 1000.0

9.5.2 4000.0 2000.0

9.5.3 0.0 0.0

9.6.calculateFoodDeduction

Input: bringsOwnFood,foodDeductionAmount

Output: double

bringsOwnFood foodDeductionAmount expectedResult

9.6.1 true 50.0 0.0

9.6.2 false 50.0 50.0

9.6.3 false 0.0 0.0

9.7. calculateTotalIncome

Input1: basicSalary,overtimeHoursValue,bonuses

Output: double

basicSalary overtimeHoursValue bonuses expectedResult

9.7.1 2000.0 200.0 100.0 2300.0

9.7.2 4000.0 0.0 500.0 4500.0

9.7.3 0.0 0.0 0.0 0.0

9.8. calculateTotalExpenses

Input: iessContribution,biweeklyAdvance,iessLoans,companyLoans,fines,foodDeduction

Output: double

iessContribution biweeklyAdvance iessLoans companyLoans fines foodDeduction expectedResult

9.8.1 200.0 1000.0 100.0 50.0 20.0 50.0 1420.0

9.8.2 400.0 2000.0 0.0 0.0 0.0 0.0 2400.0

9.8.3 0.0 0.0 0.0 0.0 0.0 0.0 0.0

9.9. calculateNetPayment

Input: totalIncome, totalExpenses

Output: double

totalIncome totalExpenses expectedResult

9.9.1 5000.0 3000.0 2000.0

9.9.2 3000.0 3500.0 -500.0

9.9.3 0.0 0.0 0.0

9.10. calculateEmployerContribution

Input: totalIncome, reserveFunds

Output: double

totalIncome reserveFunds expectedResult

9.10.1 5000.0 416.5 658.91

9.10.2 3000.0 249.9 395.34

9.10.3 0.0 0.0 0.0

9.11. calculateTotalEmployeeCost

Input: netPayment, employerContribution

Output: double

netPayment employerContribution expectedResult

9.11.1 2000.0 658.91 2658.91

9.11.2 -500.0 395.34 -104.66

9.11.3 0.0 0.0 0.0

MainWindow

1.1 Main

Input: String[] args

Output: void

args expectedResult

[] Programa ejecutado correctamente, muestra menÃº principal

["test"] Programa ejecutado correctamente, muestra menÃº principal

UserInterface

2.1. showMainMenu

Input: User input (simulated for testing)

Output: void (based on user input)

Scenario expectedResult

Input 1 addEmployee()-called

Input 2 viewEmployees()-called

Input 3 editEmployee()-called

Input 4 deleteEmployee()-called

Input 5 generatePayrolls()-called

Input 6 Program exits

Invalid input Error message, menu shown again

2.2. viewEmployee

Input: None

Output: void

Scenario expectedResult

Employees in list Employee information printed correctly

No employees in list "No hay empleados registrados." printed

2.3. addEmployee

Input: Datos del nuevo empleado

Output: void

Datos del empleado expectedResult

Datos vÃ¡lidos Empleado agregado correctamente, tamaÃ±o de la lista aumenta en 1

Datos invÃ¡lidos (campos vacÃ­os) Error mostrado, tamaÃ±o de la lista no cambia

2.4. editEmployee

Input: Ãndice del empleado, nuevos datos

Output: void

Ãndice Nuevos datos expectedResult

VÃ¡lido Datos vÃ¡lidos Empleado actualizado correctamente

InvÃ¡lido Datos vÃ¡lidos Error mostrado, ningÃºn empleado actualizado

VÃ¡lido Datos invÃ¡lidos Error mostrado, empleado no actualizado

2.5. deleteEmployee

Input: Ãndice del empleado a eliminar

Output: void

Ãndice expectedResult

VÃ¡lido Empleado eliminado correctamente, tamaÃ±o de la lista disminuye en 1

InvÃ¡lido Error mostrado, tamaÃ±o de la lista no cambia

2.6. getUserInput

Input: String-errorMessage

Output: String

Scenario expectedResult

Valid input Input returned as lowercase, trimmed string

Empty input Error message printed, empty string returned

2.7. getBooleanInput

Input: String-errorMessage

Output: boolean

Scenario expectedResult

Input "true" true returned

Input "false" false returned

Invalid input Error message printed, false returned

2.8. getIntInput

Input: String-errorMessage

Output: int

Scenario expectedResult

Valid integer Integer value returned

Non-integer Error message printed, 0 returned

EmployeeManager

3.1. EmployeeManager

Input: index, employee

Output: void

index employee expectedResult

0 {name: "John", lastName: "Doe Updated", ...} Empleado actualizado correctamente

1 {name: "Jane", lastName: "Smith Updated", ...} Empleado actualizado correctamente

-1 {name: "Invalid", ...} Error: Ã­ndice invÃ¡lido

100 {name: "OutOfBounds", ...} Error: Ã­ndice fuera de rango

3.2. addEmployee

Input: employee

Output: void

employee expectedResult

Employee("John", "Doe", "1234", Date(), 460, 10, 0, 50, 0, 0, 0, true) Employee added successfully

Employee("Jane", "Smith", "5678", Date(), 460, 5, 1, 100, 50, 0, 10, false) Employee added successfully

3.3. updateEmployee

Input: index, employee

Output: void

index employee expectedResult

0 Employee("John", "Doe Updated", "1234", Date(), 460, 15, 0, 75, 0, 0, 0, false) Employee updated successfully

-1 Any valid employee Error: Invalid index

100 Any valid employee Error: Index out of bounds

3.4. removeEmployee

Input: int index

Output: void

Scenario expectedResult

Valid index Employee removed, list size decreased by 1

Invalid index Error message printed, list unchanged

3.5. getEmployees

Input: None

Output: List<Employee>

Scenario expectedResult

After adding employees List<Employee> with all added employees

No employees added Empty List<Employee>

3.6. loadEmployees

Input: None

Output: List<Employee>

Scenario expectedResult

Valid "employees.json" file List<Employee> with correct data

Empty "employees.json" file Empty List<Employee>

File doesn't exist Empty List<Employee>

3.7. saveEmployees()

Input: None (uses internal employees list)

Output: void

Scenario expectedResult

List with employees File "employees.json" created/updated with correct content

Empty employees list Empty file created

LoginScreen

4.1. registerUser

Input: None (username and password are input during method execution)

Output: void

Input during execution expectedResult

username: "newuser", pass: "pass123" Usuario registrado exitosamente

username: "admin", pass: "newpass" El usuario ya existe

username: "", pass: "pass123" Error: nombre de usuario vacÃ­o

username: "newuser", pass: "" Error: contraseÃ±a vacÃ­a

4.2. checkPassword

Input: None (username and password are input during method execution)

Output: boolean

Input during execution expectedResult

username: "admin", pass: "admin" true

username: "admin", pass: "wrong" false

username: "notuser", pass: "pass123" false

PayrollGenerator

5.1. generatePayrolls

Input: List<Employee> employees

Output: void

employees expectedResult

List with multiple valid employees Payrolls generated and saved successfully

Empty list No payrolls generated, appropriate message shown

List with one employee with invalid data Error handled, other valid payrolls generated

5.2. generateIndividualPayroll

Input: employee, startDate, endDate

Output: IndividualPayroll

employee startDate endDate expectedResult

Employee(460, 10, 100, 50, 0, 0, false) 2023-07-01 2023-07-31 IndividualPayroll object with correct calculations

Employee(2000, 0, 0, 0, 100, 50, true) 2023-07-01 2023-07-31 IndividualPayroll object with correct calculations

null 2023-07-01 2023-07-31 Error: null employee

5.3. generateGeneralPayroll

Input: List<Employee> employees

Output: List<GeneralPayroll>

Scenario expectedResult

List with valid employees List<GeneralPayroll> with correct data for each employee

Empty list Empty List<GeneralPayroll>

null list Error handled, empty List<GeneralPayroll> returned

5.4. savePayrollsToFile

Input: JsonObject payrollsJson

Output: void

Scenario expectedResult

Valid payrollsJson File "payrolls.json" created/updated with correct content

Empty payrollsJson Empty file created

null payrollsJson Error handled, no file created/updated

IndividualPayroll

6.1. calculateTotalIncome

Input: None (uses class properties)

Output: double

Scenario expectedResult

Normal values Correct total income calculated

Zero values 0.0 returned

Very large values Correct large total income calculated

IndividualPayroll

7.1. getStartMonthDate

Input: None

Output: Date

expectedResult: First day of current month at 00:00:00

7.2. getEndMonthDate

Input: None

Output: Date

expectedResult: Last day of current month at 23:59:59.999

Income

8.1. calculateTotalExpenses

Input: None (uses class properties)

Output: double

Scenario expectedResult

Normal values Correct total expenses calculated

Zero values 0.0 returned

Very large values Correct large total expenses calculated